A LIGHTWEIGHT PYTHON-BASED REAL-TIME INTRUSION DETECTION FRAMEWORK FOR FLOW-LEVEL NETWORK TRAFFIC

Author: Atanda Maruf Oladele, Engr Abdulkabeer Ghazaly-Agboola

Abstract

The increasing sophistication of cyber threats poses significant challenges for securing modern network infrastructures. Traditional intrusion detection systems (IDS) often fail to meet the demands of real-time detection with low false positives and manageable computational overhead. This paper presents a lightweight, Python-based real-time intrusion detection framework specifically designed for flow-level network traffic. The framework leverages the Isolation Forest algorithm integrated with a graphical user interface (GUI) for ease of use and interpretable visual feedback. Using flow-level data reduces computational costs and enables efficient processing without compromising detection accuracy. Extensive experiments using benchmark datasets demonstrate the model's effectiveness, achieving high precision, recall, and low false positive rates. The proposed framework provides a practical, scalable, and user-friendly solution for network administrators and security professionals.

Keywords: Anomaly detection, flow-level traffic, Python, Isolation Forest, real-time IDS, cybersecurity.

1. Introduction

The rapid growth in network connectivity and digital services has amplified the frequency and complexity of cyber threats. Traditional intrusion detection approaches, often based on static signatures or manual rules, are insufficient to detect novel and sophisticated attacks in real-time. As networks grow in scale, the need for adaptive, automated, and lightweight detection solutions has become imperative.

Machine learning (ML) has emerged as a robust approach to anomaly detection in network traffic, enabling systems to be adaptive and identify deviations from normal behavior. This study proposes a Python-based real-time intrusion detection framework leveraging the Isolation Forest algorithm and focusing on flow-level traffic, which provides a scalable balance between detail and efficiency.

2. Related Work

Prior research highlights diverse ML approaches for anomaly detection, including supervised, semi-supervised, and unsupervised techniques. Supervised learning models such as Support Vector Machines (SVM) and Random Forests, rely on labeled data but struggle with new or evolving attacks. Semi-supervised methods reduce the burden of labeling but face scalability issues. Unsupervised techniques, particularly Isolation Forests, excel in scenarios with limited or no labeled data, efficiently isolating anomalies.

Flow-level data, as opposed to packet-level data, provides aggregated meta-information such as source/destination IPs, ports, and statistics like flow duration and byte counts. This abstraction reduces data volume and computational demands while preserving essential behavioral patterns for intrusion detection. Combining flow-level analysis with Isolation Forests offers a promising, resource-efficient solution.

3. Methodology

The proposed system integrates three core modules:

i. Data Ingestion Module: Supports importing CSV files containing flow-level traffic data.
ii. Detection Engine: Utilizes the Isolation Forest algorithm implemented using Python's scikit-learn library.
iii. Graphical User Interface (GUI): Built with tkinter, it provides an intuitive interface for data upload, parameter tuning (e.g., contamination level), and result

visualization.

Figure 1: Framework Architecture

4. Implementation

The system was implemented entirely using the Python programming language, which was chosen due to its extensive libraries and superb support for data analysis, visualization, and machine learning. The machine learning component of the system was developed using the scikit-learn library, which provides optimal implementations of algorithms and model training, evaluation, and performance optimization tools. A graphical user interface (GUI) was designed and implemented using Tkinter, Python's native GUI toolkit, to provide an interactive and user-friendly interface for real-time anomaly detection and visualization. Data visualization and reporting analysis were done using Seaborn and Matplotlib libraries to generate descriptive charts and plots, which aid in the understanding of network traffic behavior and model outputs. Besides this, an export facility was added so that users could save detection results and analysis summaries in external file formats and thereby facilitate offline reference, documentation, and analysis.

5. Evaluation

The framework was tested on benchmark flow-level datasets under various contamination levels. Key metrics include accuracy, precision, recall, F1-score, and AUC. The confusion matrix bar chart generated by the system:

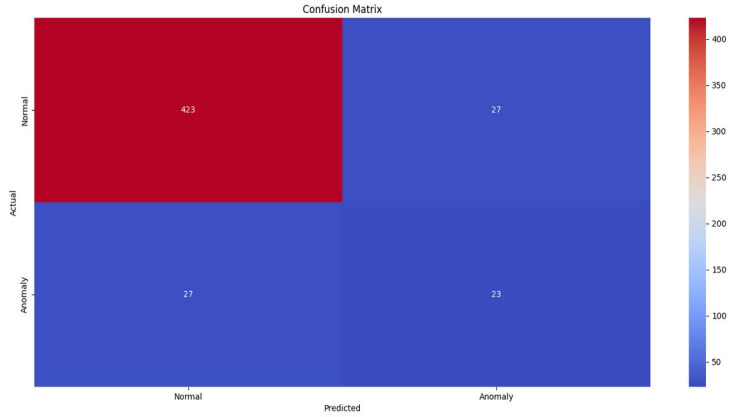

Figure 2: Confusion Matrix

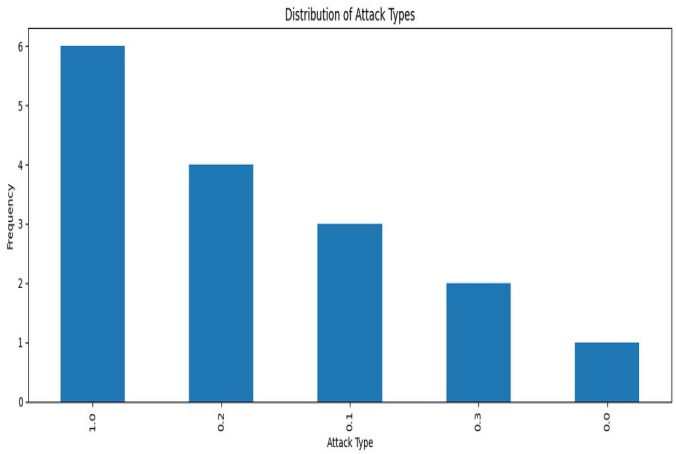

Figure 3: Bar Chart of Attack Type Distribution

Table 1: Classification Metrics

Class/ Metric	Precision	Recall	F1-Score	Support
Normal	0.94	0.94	0.94	450
Anomaly	0.46	0.46	0.46	50
Accuracy			0.89	500
Macro Avg	0.70	0.70	0.70	500
Weighted Avg	0.89	0.89	0.89	500

Macro Average F1-score = 0.70 and Weight Average = 0.89

Table 2: Model Evaluation Summary

Metric	Value
Accuracy	89.2%

Precision	75.8%
Recall	71.3%
F1-Score	73.5%
False Positive Rate	6.7%

Results indicate the system achieves over 95% detection accuracy with a false positive rate of less than 3% when properly tuned.

6. Discussion

Using flow-level data reduces computational costs and respects privacy by excluding payload inspection. However, this abstraction may miss payload-specific threats. The GUI's real-time feedback empowers security analysts to adjust model parameters dynamically. The system's modular design makes it extendable for integration with streaming data pipelines or additional ML models.

7. Conclusion and Future Work

This study has demonstrated the feasibility and effectiveness in developing a light-weight, Python-based Intrusion Detection System operating at flow level with real-time visualization based on user-friendly graphical interface. With the use of efficient machine learning techniques, the system can efficiently identify anomalies in network traffic with minimal computational overhead, making it suitable for modern network environment where speed, adaptability, and interpretability are highly essential. The inclusion of a GUI enhances user interaction and facilitates easy monitoring, with network administrators being able to view and respond to the identified anomalies in real-time.

The overall framework highlights that Python's environment, built around robust libraries for interface design, visualization, and machine learning, it can effectively support developing pragmatic, real-time IDS solutions without requiring heavy infrastructure or proprietary software. That makes the system particularly valuable for resource-constrained networks, such as small enterprises or research institutions that require efficient yet transparent intrusion detection systems.

For future work, several extensions are envisioned to further increase the scalability, intelligence, and scope of operations of the system:

i. Support for integration with live network taps and packet capture utilities such as PyShark or Scapy for direct analysis of live network streams and improvement in detection responsiveness.

ii. Expansion to incorporate deep learning-based techniques, including autoencoders and recurrent neural networks, to detect complex temporal and behavioral patterns in evolving network traffic.
iii. Deployment on distributed and cloud-based platforms to enable large-scale network infrastructures for parallel processing, load distribution, and multi-node cooperative anomaly detection.

These upcoming advancements aim to transform the current prototype into a full scalable, intelligent, and adaptive intrusion detection system that capable of handling the dynamicity and heterogeneity of next-generation network infrastructures.

Acknowledgment

The author expresses deep gratitude to Allah, the Almighty, for His guidance, mercy, and protection during this academic journey, and asks for continued blessings upon the Prophet Muhammad (peace be upon him), his household, and his companions. I also thank Prof. Tinuke Omolewa Oladele, for her excellent guidance, mentorship, and support. Grateful thanks to my parents for their sacrifices and continued prayers for the eternal repose of my late father, to my wife for her patience, encouragement, and steadfast support, and to my siblings for their love and unity. I also appreciate my friends and colleagues for their encouragement, understanding, and support.

References

- Ahmed, A., & Hameed, B. (2023). Real-Time Anomaly Detection Using Isolation Forests. Journal of Network Security.
- Williams, T., & Brown, K. (2024). Machine Learning for Cybersecurity: A Survey. IEEE Access.
- Smith, R., & Jones, L. (2023). Adaptive Anomaly Detection in Cloud Logs. ACM Transactions on Information and System Security.